## REMARKS

Claims 1, 13, 14, 37, 71 and 94 are amended. Claims 62-63 were previously cancelled. Claims 1-61 and 64-96 remain in the application for consideration. In view of the following remarks, Applicant respectfully requests reconsideration and allowance of the subject application.

### 35 U.S.C. § 112 Rejections

Claim 94 stands rejected under 35 U.S.C. § 112, second paragraph as being indefinite. The Office states that the term "sorting the files according to average order in which the files were downloaded" is confusing as it implies that the files have already been downloaded.

Applicant has amended claim 94 to now recite that the "sorting by file usage order comprises sorting the files according to the average order in which the files were downloaded within ***scenarios of*** their particular priority or priorities." As noted in the Specification, a scenario is a script of tasks that the average user typically follows when using a product during a particular portion of product use. See, e.g. Specification, page 28, lines 22-24. This, in combination with the context of the claim (i.e. "A method for ordering files for download to a client...) should make it apparent that this claim is directed to ordering the files before download.

In view of the amendment and discussion above, Applicant respectfully submits that there is nothing indefinite with respect to claim 94.

**35 U.S.C. §§ 102 and 103 Rejections**

Claim 82 stands rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 5,999,740 to Rowley.

Claims 1-5, 8-20, 37-39, 42-46, 55, 57, 61, 62, and 71-76 stand rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view of U.S. Patent No. 6,219,698 to Iannucci and a publications entitled "Delphi 5 Developer's Guide" by Xavier Pacheco et al. (hereinafter "Pacheco").

Claims 6, 40, 41 and 54 stand rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view of Iannucci, Pacheco and U.S. Patent No. 4,641,274 to Swank.

Claim 7 stands rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view of Iannucci, Pacheco and U.S. Patent No. 5,195,183 to Miller.

Claims 21-24, 27, 28, 32, 36, 47, 49-51, 53, and 54 stand rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view Swank.

Claims 25 and 29-31 stand rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view of Swank and U.S. Patent No. 5,845,090 to Collins III.

Claim 26 stands rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view of Swank and U.S. Patent No. 6,615,276 to Mastrianni.

Claims 33-35 stand rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view of Swank and U.S. Patent No. 5,835,777 to Staelin.

Claims 48 and 52 stand rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view of Swank and Iannucci.

Claims 58 and 59 stand rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view of Iannucci, Pacheco and an article entitled "The Component Object Model: A Technical Overview" by Williams (hereinafter "Williams").

Claims 60 stands rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view of Iannucci, Pacheco and Collins III.

Claims 64, 65, 67 and 68 stand rejected under 35 U.S.C. § 103(a) as being obvious over Collins III in view of U.S. Patent No. 5,859,973 to Carpenter.

Claim 66 stands rejected under 35 U.S.C. § 103(a) as being obvious over Collins III in view of Carpenter and U.S. Patent No. 5,826,265 to Van Huben.

Claim 69 stands rejected under 35 U.S.C. § 103(a) as being obvious over U.S. Patent No. 6,282,711 to Halpern in view of U.S. Patent No. 5,721,824 to Taylor.

Claim 70 stands rejected under 35 U.S.C. § 103(a) as being obvious over Halpern in view of Taylor and Rowley.

Claims 77-80 stand rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view of Iannucci, Pacheco and U.S. Patent No. 4,910,663 to Bailey.

Claim 81 stands rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view of Iannucci, Pacheco and U.S. Patent No. 5,761,408 to Kolawa.

Claims 83 and 85 stand rejected under 35 U.S.C. § 103(a) as being obvious over Rowley.

Claim 84 stands rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view of Bailey.

Claims 86-90 stand rejected under 35 U.S.C. § 103(a) as being obvious over Bailey in view of Kolawa and Collins III.

Claims 91-93, 95 and 96 stand rejected under 35 U.S.C. § 103(a) as being obvious over Rowley in view of Collins III and Bailey.

Before discussing the substance of the Office's rejections, the following discussion is provided to assist the Office in appreciating the patentable distinctions between Applicant's claimed subject matter and the various references cited by the Office.

**The Rowley Reference**

Rowley's is perhaps best appreciated with respect to FIG. 1, which shows a computer network comprising a number of client computers 101 and a number of server computers 102 interconnected by a network 103.

Each file server 102 stores a number of application files 104, forming a number of software applications. Normally, each application consists of several application files. The application files are stored in compressed form, using any standard data compression technique.

Conveniently, the server has a number of application directories, one for each application. Each of these directories has a number of sub-directories, which hold the new or amended application files for different versions of the application. Typically, one of these versions is an installer version of the application, while the other versions are non-installers. An installer is an executable program which, when run on a client machine, sets up the correct environment for the application. Normally, the first release of an application is an installer, and subsequent releases are non-installers.

Each file server also stores a number of manifest files 105, one for each version of each application stored on the server. These manifest files are stored in

the relevant sub-directories, and each has a name constructed from the name and release number of the application to which it relates. Each manifest file contains a list of the application files that make up the particular version of the application. For each application file, it contains the following parameters:

- The filename of the application file.
- The version number of the application file.
- The target directory into which the application file should be installed.
- Date and time of issue.
- File size and compressed file size.
- An action parameter which indicates whether the file is to be installed, to be deleted, or to be executed on download.
- A flag which indicates access permissions of the file, e.g. read only.
- A cyclic redundancy checksum (CRC).

FIGS. 3A and 3B show the operation of an update program 110 (Fig. 1). Referring to FIG. 3A, the update program first contacts one of the servers 102 (Step 301), by way of the network 103, to obtain the live release file from that server. The update program then compares this release file with its locally held registration file 109 (Step 302), to identify which of the currently installed applications have more recent versions available. It also identifies any new application installers in the release file for applications that are not installed locally on the client.

The update program then displays a screen, as shown in FIG. 9, which allows the user to select either an "Updates" option or a "New Release" option. If the user selects the "Updates" option, the update program proceeds to step 303 in FIG. 3A. Alternatively, if the user selects the "New Release" option, the update program proceeds to step 310 in FIG. 3B.

If the user selects the "Updates" option (Step 303), a list of the titles and versions of currently installed applications is displayed, as shown in FIG. 9. The display indicates which, if any, of the applications have more recent versions available. The user may select from this list one or more (or all) of the applications for which a more recent version is available. Selection of an application will automatically cause any dependent applications to be selected.

If there is a more recent version available of the update program itself, this is automatically selected. Hence, every time the update program runs it will update itself if necessary.

If the user selects the "OK" button on this screen, the program proceeds to Step 304 below. Alternatively, the user may simply exit from the program without performing any updates by selecting the "Cancel" button. Assuming the "OK" button was selected in step 303, the update program contacts the server 102 to obtain the manifest file for the first (or only) of the selected applications (Step 304). The update program then determines differences between files installed on the client computer and those listed in the manifest file (Step 305). For each application file listed in the manifest file, a check is made to determine whether the specified file is already present in the specified directory in the client by using CRC checks. If not, the program contacts the server 102, to retrieve the required application file. The retrieved file is expanded, and then checked for file-transfer corruption, using the CRC checksum. All the application files are read into a temporary directory on the client computer. Thus, the update program does not fetch any application file if the required version of that file is already installed in the required directory, thereby eliminating unnecessary traffic over the network.

When all the files listed in the manifest file have been correctly retrieved (Step 306), the installation actions are implemented as follows. Any files marked for deletion are deleted from the client computer, files marked for execution are executed and files marked for installation are installed into the specified directories in the client, provided the file version is more advanced than that of the existing file. Hence then any existing files with the same names in the directories will be overwritten.

If, on the other hand, some of the file transfers failed, none of the files are installed. Instead, a message is displayed, giving the user the option of either canceling the update, or making another attempt to access the files.

If all the required applications have now been updated (Step 307), the update program proceeds to Step 308. Otherwise it returns to Step 304 above to get the manifest file for the next required application to be updated.

**The Iannucci Reference**

Iannucci discloses a system for sending and receiving automatic message notification and remote client configuration. As shown in Iannucci's Fig. 1, a server 100, a client 110 and at least one message server 120 communicate with each other via a communication network 130.

The server 100 stores a software application 159 and a database 155. The database 155 has a current version number 158 of a particular software application, which will be called "application A" and which may be stored as part of the programming 159, a list of clients 157, and information 160 relating to the availability of a message including persistent state information.

The client 110 includes a processor 170 having an elapsed time counter 171, a display 180 and a memory 185. The memory 185 stores a database 190 and software 187, including the version of software "application A" which presently operates on the processor 170. The client 110 communicates with the server 100 via the communications network 130 and is capable of directing the display of web pages 195 and 197 on the display 180.

An accept/reject button 191, a message waiting indicator 192, and a configuration or upgrade message 193 and client state change directives 198 are shown displayed within the web page 195 on the display 180. The button 191, configuration message 193, and the directives 198 are all part of the configuration information 196 which appears within web page 195. The message 196 appears in a separate web page 197.

The database 190 contains server Universal Resource Locators (URL's) including the URL 210 associated with the server 100, a first persistent state value A 230 and a second persistent state value B 240, message URL's including the URL 250 associated with the message server 120 and a frequency time 260. If available, the frequency time is a user selected minimum time period between upgrade availability notifications.

Referring additionally now to FIG. 2, in response to a client processor 170 initializing the "application A" software stored on memory 185, the server 100, via the communication network 130, automatically receives a signal in step 300 which represents status information from the client 110, in accordance with programmed instructions included in the software 187 stored on the client memory 185. Among other information, the status information includes a client version number of the software "application A" stored on memory 185. Software "application A" may

have been originally downloaded to the client 110 from server 100, or from a different network server. Alternatively, software "application A" could have been loaded directly to the client 110 from a floppy disk or other physical storage medium.

The server 100, in step 320, compares the client version number of software "application A" stored in client 110 against a current version number 158 of the software "application A" stored in the memory 155 of server 100 and determines whether the current version number 158 is greater than the client version number. If so, the server processor 140, in accordance with programmed instructions which form part of the software 159 stored on server memory 105, generates configuration information 196 in step 330. The configuration information 196 includes a configuration message 193 which contains information pertaining to features of the available upgrade of the software "application A" to current version number 158, and an offer to download the software application A upgrade. The configuration information 196 also includes, an accept/reject button 191. The configuration information 196 may further include client persistent state change directives 198. The server processor 140, in accordance with its programmed instructions, also directs the transmission of the configuration information 196 via the network 130 to the client 110 in step 330. The server processor 140, in step 350, additionally determines whether the offer is accepted or rejected through the receipt of a new request from client 110 to download the upgrade. If the offer is accepted the server 100 first downloads a web page containing a link to the software upgrade. Responsive to the user clicking on the link, the server 100 downloads the new version of the software to client 110 in step 360.

The configuration information 196, including the configuration message 193 and the accept/reject button 191 are displayed on the client display 180 in step 460 of FIG. 3. By clicking on the displayed accept/reject button 191 to accept the upgrade of software "application A", a signal is generated and communicated from the client processor 170 to the server processor 140 via the network 130, responsive to which the server processor 140, in step 360, directs the downloading of the web page containing the link to the upgrade to the client 110. If accepted, the upgrade is downloaded and stored by processor 170 on client memory 185. By clicking on the accept/reject button 191 to accept or reject the upgrade, the configuration information 196 will be eliminated from the display 180 as indicated in step 370. Until the button 191 is clicked on, the configuration information 196 will continue to be displayed during the current session and will be redisplayed during each future session, subject to the selected minimum time periods between upgrade notification. If the upgrade is accepted or rejected, the previously displayed configuration information 196 will not be displayed during future sessions.

**Claims Rejected under § 102 over Rowley and Related Dependent Claims Rejected Under § 103**

**Claim 82** recites a method for creating a package manifest comprising:

- receiving information pertaining to one or more extension directories that contain files that comprise a software extension that is to be incorporated into a software application program to extend the application program;
- receiving, if any, information pertaining to file groups or load dependencies;
- receiving, if any, information pertaining to file usage statistics; and
- generating a package manifest based on the received information.

In making out the rejection of this claim, the Office argues that Rowley anticipates this claim's subject matter. Specifically, the Office cites to Rowley's column 4, lines 57-61 and Fig. 8 as disclosing subject matter that anticipates this claim. Applicant respectfully disagrees and traverses the rejection.

For example, the Office argues that Rowley discloses receiving information that pertains to file usage statistics and cites to Fig. 8 and argues that Rowley's flag indicating permissions of the file is a "file usage statistic." In addition, in addressing Applicant's previous arguments, the Office states that a "read only" flag ***is*** a statistic because it dictates ***how*** a file is used. See, e.g. Office Action, page 20, first full paragraph.

Applicant respectfully disagrees. Rowley's flag is provided to mark a particular file as a "read only" file. Applicant submits that there is nothing *statistical* about a flag that marks a file as a "read only" file as the term "file usage statistic" is used in the Specification.

As an example, consider the following material from the Specification starting on page 28, line 20 through page 30, line 6:

> The ***file usage statistics*** from scenario runs parameter...enables the file download priority to be determined based on scenario runs. A scenario is a script of tasks that the average user typically follows when using a product during a particular portion of product use. For example, one scenario might pertain to the tasks involved in sending an email message (i.e. click "new mail" button, type in "TO" well, type is "Subject" well, etc.). In the described embodiment, file usage statistics from scenario runs are collected from running IIS logs on various scenarios. ***The different scenarios are directed to ensuring, with some degree of probabilistic support, that the file download order reflects, in some way, the files that will likely be used by the user first.***

Hence, it should be quite clear that Rowley's "read only" flag neither discloses nor suggests file usage statistics as that term is utilized in the claim and the Specification. Accordingly, for at least this reason, this claim is allowable.

**Claims 83-85** depend from claim 82 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 82, are neither disclosed nor suggested in the references of record, either singly or in combination with one another. In addition, given the allowability of claim 82, the rejection of claim 84 over the combination with Bailey is not seen to add anything of significance.

**The § 103 Standard**

To establish a prima facie case of obviousness, three basic criteria *must* be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992); *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988). Second, there must be a reasonable expectation of success. *In re Merck & Co., Inc.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986). Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974).

Hence, when patentability turns on the question of obviousness, the search for and analysis of the prior art includes evidence relevant to the finding of whether there is a teaching, motivation, or suggestion to select and combine the

references relied on as evidence of obviousness. See, e.g., *McGinley v. Franklin Sports, Inc.*, 262 F.3d 1339, 1351-52, 60 USPQ2d 1001, 1008 (Fed. Cir. 2001) ("the central question is whether there is reason to combine [the] references," a question of fact drawing on the Graham factors).

"The factual inquiry whether to combine references must be thorough and searching." *Id.* It must be based on objective evidence of record. This precedent has been reinforced in myriad decisions, and cannot be dispensed with. See, e.g., *Brown & Williamson Tobacco Corp. v. Philip Morris Inc.*, 229 F.3d 1120, 1124-25, 56 USPQ2d 1456, 1459 (Fed. Cir. 2000) ("a showing of a suggestion, teaching, or motivation to combine the prior art references is an 'essential component of an obviousness holding'") (quoting *C.R. Bard, Inc., v. M3 Systems, Inc.*, 157 F.3d 1340, 1352, 48 USPQ2d 1225, 1232 (Fed. Cir. 1998)); *In re Dembiczak*, 175 F.3d 994, 999, 50 USPQ2d 1614, 1617 (Fed. Cir. 1999) ("Our case law makes clear that the best defense against the subtle but powerful attraction of a hindsight-based obviousness analysis is rigorous application of the requirement for a showing of the teaching or motivation to combine prior art references."); *In re Dance*, 160 F.3d 1339, 1343, 48 USPQ2d 1635, 1637 (Fed. Cir. 1998) (there must be some motivation, suggestion, or teaching of the desirability of making the specific combination that was made by the applicant); *In re Fine*, 837 F.2d 1071, 1075, 5 USPQ2d 1596, 1600 (Fed. Cir. 1988) ("'teachings of references can be combined only if there is some suggestion or incentive to do so.'") (emphasis in original) (quoting *ACS Hosp. Sys., Inc. v. Montefiore Hosp.*, 732 F.2d 1572, 1577, 221 USPQ 929, 933 (Fed. Cir. 1984)).

The need for specificity pervades this authority. See, e.g., *In re Kotzab*, 217 F.3d 1365, 1371, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000) ("particular findings

must be made as to the reason the skilled artisan, with no knowledge of the claimed invention, would have selected these components for combination in the manner claimed").

**Claims Rejected Over the Combination that Includes at least Rowley and Iannucci**

**Claim 1** has been amended and recites a method for delivering software via a network comprising [added language appears in bold italics]:

- describing one or more software extensions using a hierarchical language, the extensions being configured for incorporation on a client, said describing defining one or more manifests containing at least one list of files comprising an extension; and
- delivering the one or more manifests to the client via the network, the one or more manifests being configured for use in downloading the software extensions via the network, at least some of the extensions being downloadable by streaming extension files to the client in a manner that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load, ***said manner being developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that are less likely to be used first can be downloaded via a background download process.***

In making out the rejection of this claim, the Office argues that Rowley teaches the recited acts of describing and delivering, except for describing the extensions using a hierarchical language. The Office then relies on Iannucci and argues that it teaches a manifest file in the form of a web page which is sent to the client for use in downloading a software extension. In addition, responsive to the previous amendment, the Office has added Pacheco to the combination and argues

that Pacheco teaches downloading "any type" of data to a client using streaming—which allows users to interact with the files while the file is downloading.

Based on this, the Office argues that this claim would be obvious in view of the combination of Rowley, Iannucci and Pacheco.

Applicant respectfully disagrees and maintains its position with respect to the Office's interpretation and application of Iannucci. Nonetheless, Applicant has amended claim 1 to recite that the "manner" that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load is ***developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that are less likely to be used first can be downloaded via a background download process***. Support for this amendment can be found in the Specification on page 18, lines 18-24.

Pacheco, on which the Office relies for its streaming teaching, generally discusses the notion of data streaming. Pacheco does not teach or suggest scenario runs in which files that are more likely to be first used by a user are downloaded before other files. Further, Pacheco does not teach or suggest downloading other such files via a background download process.

Accordingly, for at least this reason, claim 1 is allowable.

**Claims 2-12** depend from claim 1 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 1, are neither disclosed nor suggested in the references of record, either singly or in combination with one another. In addition, given the allowability of claim 1 over the combination of Rowley and Iannucci, the rejections of claim 6 over the further combination with

Swank, and of claim 7 over the further combination with Miller are not seen to add anything of significance.

**Claim 13** recites one or more computer-readable media comprising computer-readable instructions thereon which, when executed by a computer, cause the computer to [added language appears in bold italics]:

- describe one or more software extensions using extensible markup language (XML), the extensions being configured for incorporation on a client, said describing defining a manifest containing at least one list of files comprising an extension, the manifest being configured to assist in one or more of the following: organizing delivery of individual files listed in the manifest, validating individual files listed in the manifest, and updating individual files listed in the manifest; and
- deliver the manifest to the client via the network, at least some of the extensions being downloadable by streaming extension files to the client in a manner that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load, ***said manner being developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that are less likely to be used first can be downloaded via a background download process.***

In making out the rejection of this claim, the Office argues that the combination of Rowley, Pacheco and Iannucci render the subject matter of this claim obvious. Applicant respectfully disagrees with the Office's combination and rationale. Nonetheless, Applicant has amended this claim to recite that the "manner" that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load is ***developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that***

***are less likely to be used first can be downloaded via a background download process***. Support for this amendment can be found in the Specification on page 18, lines 18-24.

Pacheco, on which the Office relies for its streaming teaching, generally discusses the notion of data streaming. Pacheco does not teach or suggest scenario runs in which files that are more likely to be first used by a user are downloaded before other files. Further, Pacheco does not teach or suggest downloading other such files via a background download process.

Accordingly, for at least this reason, this claim is allowable.

**Claim 14** has been amended and recites a method for receiving software via a network comprising [added language appears in bold italics]:

- receiving a manifest that contains at least one list of files comprising a software extension that is to be downloaded via a network and incorporated on a client, the manifest being defined in extensible markup language (XML), the manifest being configured to assist in:
  - organizing delivery of the files,
  - validating individual files listed in the manifest, and
  - updating individual files listed in the manifest; and
  - downloading files from the list of files contained in the manifest;
- wherein the extension is downloadable by streaming extension files to the client in a manner that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load, ***said manner being developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that are less likely to be used first can be downloaded via a background download process***.

In making out the rejection of this claim, the Office makes the same argues that it made with respect to claim 13 above. Applicant disagrees with the Office's

rejection and rationale. Nonetheless, Applicant has amended this claim to recite that the "manner" that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load is ***developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that are less likely to be used first can be downloaded via a background download process***. Support for this amendment can be found in the Specification on page 18, lines 18-24.

Pacheco, on which the Office relies for its streaming teaching, generally discusses the notion of data streaming. Pacheco does not teach or suggest scenario runs in which files that are more likely to be first used by a user are downloaded before other files. Further, Pacheco does not teach or suggest downloading other such files via a background download process.

Accordingly, for at least this reason, this claim is allowable.

**Claims 15-20** depend from claim 14 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 14, are neither disclosed nor suggested in the references of record, either singly or in combination with one another.

**Claim 37** has been amended and recites a method of providing software via a network comprising [added language appears in bold italics]:

- describing one or more software extensions using one or more extensible markup language (XML) files, the extensions being configured for incorporation in a software program executing on a client, individual XML files providing individual manifests that contain a list of files that comprise an extension; and

- storing the XML files in a Web-accessible location;
- wherein at least some of the extensions are downloadable by streaming extension files to the client in a manner that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load, ***said manner being developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that are less likely to be used first can be downloaded via a background download process.***

In making out the rejection of this claim, the Office argues that the combination of Rowley, Pacheco and Iannucci renders the subject matter of this claim obvious. Applicant respectfully disagrees with the Office's combination and rationale. Nonetheless, Applicant has amended this claim to recite that the "manner" that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load is ***developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that are less likely to be used first can be downloaded via a background download process***. Support for this amendment can be found in the Specification on page 18, lines 18-24.

Pacheco, on which the Office relies for its streaming teaching, generally discusses the notion of data streaming. Pacheco does not teach or suggest scenario runs in which files that are more likely to be first used by a user are downloaded before other files. Further, Pacheco does not teach or suggest downloading other such files via a background download process.

Accordingly, for at least this reason, this claim is allowable.

**Claims 38-46** depend from claim 37 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 37, are neither disclosed nor suggested in the references of record, either singly or in combination with one another. Additionally, given the allowability of claim 37, the rejections of claims 40 and 41 over the further combination with Swank is not seen to add anything of significance.

**Claim 55** recites a data structure embodied on a computer-readable medium comprising:

- one or more first tags indicative of associated file groups associated with an Internet-downloadable software extension that can extend an application program executing on a client; and
- one or more second tags indicative of specific files that comprise the software extension.

In making out the rejection of this claim, the Office argues that Rowley teaches a manifest file which stores a list of files utilized in a software extension and further teaches one or more files groups associated with the files. The Office admits that Rowley does not teach that the individual files and file groups comprise tags that indicate individual files and file groups. The Office then relies on Iannucci and argues that it teaches a manifest in the form of a web page, which is sent to the client for use in downloading a software extension. The Office then reasons that web pages are designed using HTML—a tag-based language—and that using tags to separate fields in a web page is an inherent aspect of HTML. Based on this, the Office argues that the subject matter of this claim is obvious in

view of Rowley and Iannucci since the combination allows the manifest to take the form of a web page.

In addition, the Office addresses Applicant's previous arguments traversing this rejection on page 18, last paragraph of the present Office Action. Specifically, the Office argues that Rowley teaches that files are separated into specific directories and target directories which are file groups.

Applicant respectfully disagrees. First, the portion of Iannucci cited by the Office in support of this rejection pertains simply to a web page with a ***clickable link*** that enables a user to click on the link and subsequently download software. Second, and perhaps more importantly, there is no disclosure in either reference to provide a data structure that associates one or more first tags with *file groups* ***and*** one or more second tags with *specific files*. Rather, Rowley's manifest appears to simply list application files that make up a particular version of an application. More specifically, the parameters that are included in Rowley's manifest include a file name of an application file, a version number, a target directory, a date and time of issue, and additional information—none of which is disclosed to comprise file groups. Additionally, Iannucci simply discloses a ***clickable link***. There is no teaching or suggestion that such link is defined by or defines ***both*** first tags indicative of associated file groups ***and*** second tags indicative of specific files that comprise a software extension.

For at least these two reasons, the Office has failed to establish a *prima facie* case of obviousness and this claim is allowable.

**Claims 56-61** depend from claim 55 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 55, are neither disclosed

nor suggested in the references of record, either singly or in combination with one another. In addition, given the allowability of claim 55, the rejections of claims 58-59 over Williams, and of claim 60 over Collins, is not seen to add anything of significance.

**Claim 71** has been amended and recites an automated software tool comprising a package manifest creation tool configured to [added language appears in bold italics]:

- receive one or more input parameters pertaining to a package manifest that is to describe a software extension that is configured to extend a software application executing on a client; and
- generate a package manifest that describes the extension, the package manifest being generated using a hierarchical language;
- wherein the extension is downloadable by streaming extension files to the client in a manner that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load, ***said manner being developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that are less likely to be used first can be downloaded via a background download process.***

In making out the rejection of this claim, the Office argues that Rowley teaches a manifest editor (Fig. 8) as claimed, but fails to teach that the manifest is described using a hierarchical language. The Office then relies on Iannucci to argue that the subject matter of this claim is obvious. In addition, the Office admits that neither Rowley nor Iannucci teach that extensions are downloadable by streaming the extension files to the client in a manner that enables a user to begin to interact with the extension sooner than if the user had to wait for the

entire extension to load. To supply this subject matter, the Office relies on Pacheco.

Applicant respectfully disagrees with the Office's combination and rationale. Nonetheless, Applicant has amended this claim to recite that the "manner" that enables a user to begin to interact with the extension sooner than if the user had to wait for the entire extension to load is ***developed based on scenario runs in which files that are more likely to be first used by the user are downloaded before files that are less likely to be first used, and wherein files that are less likely to be used first can be downloaded via a background download process***. Support for this amendment can be found in the Specification on page 18, lines 18-24.

Pacheco, on which the Office relies for its streaming teaching, generally discusses the notion of data streaming. Pacheco does not teach or suggest scenario runs in which files that are more likely to be first used by a user are downloaded before other files. Further, Pacheco does not teach or suggest downloading other such files via a background download process.

Accordingly, for at least this reason, this claim is allowable.

**Claims 72-81** depend from claim 71 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 71, are neither disclosed nor suggested in the references of record, either singly or in combination with one another. In addition, given the allowability of claim 71, the rejections of claims 77 and 81 over the references to Bailey and Kolawa respectively, are not seen to add anything of significance.

**Claims Rejected Over the Combination that Includes at least Rowley and Swank**

**Claim 21** recites a data structure comprising:

- a list of one or more files that are utilized in a software extension that is configured to extend a software application executing on a client;
- one or more hashes each of which being associated with a particular listed file; and
- one or more file groups, individual files being associated with individual file groups, the file groups determining when particular files of the extension get downloaded to the client;
- ***the data structure being configured to assist in delivering software extensions via the Internet.***

In making out the rejection of this claim, the Office argues that Rowley teaches a manifest file that stores a list of files and further teaches one or more file groups associated with files. The Office further argues that Rowley teaches selecting certain groups of files for downloading to form an application program and hence, determining when the files are downloaded based on which groups the files are in, since selecting a group of files determines that the groups of files is downloaded to the client.

The Office attempts to clarify its argument in its "Response to Arguments" section on page 18, first full paragraph. Specifically, the Office notes that it is true that Rowley's file groups determine *what* particular files of the extension get downloaded. The Office then extends this by arguing that Rowley's file groups allow users to choose file groups to download and hence, *when* the groups get downloaded. This logic is misplaced for at least the following reason. In Rowley, it is the *user* who determines when the files are downloaded. ***This claim***

***specifically recites that the file groups determine when particular files of the extension get downloaded.*** This is not to state or imply that the user is not involved in the process. Quite to the contrary, Applicant's disclosure provides examples of situations in which the user is involved in the download process. What this claim recites, though, is that the file groups determine ***when*** particular files get downloaded.

The Office is respectfully referred to Applicant's Specification, starting on page 19, line 15 where an exemplary implementation of a file group is described. The discussion appearing in this portion of the Specification is provided below for the Office's convenience:

> All files in an extension can be labeled according to a number of predefined file groups. The file group of a particular file determines ***when the particular file gets downloaded***, where it is stored on the client, and how it gets packaged. In the described embodiment, four predefined file groups are provided and are listed and described in the table immediately below:

| Group name | When downloaded | Where stored on the client | Packaging | Content |
|---|---|---|---|---|
| Required | Downloaded before any other files in the extension. | NetDocs package cache | All required files in an extension are packaged together as a CAB* file. | DLLs included so that a user will not have to wait for a prolonged period of time before clicking on a UI element |
| Offline | Offline files start getting downloaded as soon as Required are down. Providing the user stays on line long enough, these files will all get downloaded and will later be available for offline use. | NetDocs package cache | File are sent down individually. | Bulk of the UI files. |
| On demand | Only downloaded when they are requested for the first time. | NetDocs package cache | Files are sent down individually. | To avoid using up disk space on the client, advanced features can be put in this category. |
| Online only | Downloaded on demand. Content is only available when the user is online. | WinInet Cache | Files are sent down individually | Content that is not to be provided offline. Examples include help pages and other content that can consume a large amount of disk space. |

The Office will notice in this example, that the table describes four file group names which appear in the first column. Additionally, the second column describes when files of the group are to be downloaded. For example, "Required" files get downloaded before any other files in the extension; and "Offline" files start getting downloaded as soon as file in the "Required" group are downloaded.

Rowley neither discloses nor suggests any such subject matter. Accordingly, for at least this reason, the Office has failed to establish a *prima facie* case of obviousness and this claim is allowable.

The Office then relies on Swank, which teaches storing a hash for a file, to argue that the subject matter of this claim would be obvious. Applicant respectfully disagrees. In clarifying its position and addressing the Applicant's previous arguments, the Office states that "Swank is directed to downloading data from a server to a client using a hash to determine changes in files to determine what to download to a client." The Office then states that "[t]his is comparable to

downloading extensions from a server to a client using a hash to determine which files to download to the client." Applicant respectfully disagrees—*comparability* is not the standard that is to be used to make out an obviousness rejection.

As previously noted, Swank has nothing whatsoever to do with delivering software extensions via the Internet. Rather, Swank is directed to minimizing communications between a host computer and a personal computer by transmitting only changed lines in an updated file from a terminal to a host. Swank uses its hash processing to ascertain which text lines have been changed and hence which text lines need to be transmitted. Swank does not disclose or suggest hashes in the context of a data structure that comprises the other components recited above ***to assist in delivering software extensions via the Internet***.

It would appear that the Office's attempted combination is, at best, based on hindsight reconstruction which has been specifically proscribed by the Federal Circuit. Accordingly, for at least these additional reasons, the Office has failed to establish a *prima facie* case of obviousness and this claim is allowable.

**Claims 22-36** depend from claim 21 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 21, are neither disclosed nor suggested in the references of record, either singly or in combination with one another. In addition, given the allowability of claim 21, the rejections of claims 25 and 29-31 over Collins; of claim 26 over Mastrianni; and of claims 33-35 over Staelin is not seen to add anything of significance.

**Claim 47** recites a security method for downloading software extensions via the Internet comprising:

- receiving, via the Internet, a package manifest containing a list of multiple files that comprise a software extension that is to be incorporated into an application program executing on a client, the list containing a hash for one or more of the files comprising the software extension;
- receiving, via the Internet, the multiple files that are described in the package manifest;
- creating a hash for one or more of the multiple received files; and
- comparing the created hash of the one or more files with corresponding file hashes contained in the package manifest to ascertain whether one or more of the received file is secure.

In making out the rejection of this claim, the Office essentially maintains its position as articulated in the previous Office Action dated October 28th, 2003. In that Office Action, the Office argues that Rowley teaches receiving a package manifest and multiple files as recited in this claim. The Office admits that Rowley does not teach that the list contains a hash for one or more of the files comprising the software extension, creating a hash for one or more received files and comparing the created hash with the corresponding file hashes to ascertain the security of the file.

The Office then relies on Swank and argues that it discloses storing a hash for an individual file to be updated, creating an updated hash of a received file, and comparing the hashes to determine the integrity of the file [sic:process], citing to column 3, lines 45-51. Based on this, the Office argues that the subject matter of this claim would be obvious in view of these two references. Applicant respectfully disagrees.

The Office further addresses Applicant's arguments against this combination on page 19, first full paragraph. Specifically, the Office argues that because Swank is directed to downloading data from a server to a client, this is

comparable to downloading extensions from a server to a client. Applicant submits that *comparability* is not the obviousness standard.

As noted above, Swank has nothing whatsoever to do with receiving software extensions via the Internet. Rather, Swank is directed to improving communications by transmitting only changed lines in an updated file from a terminal to a host. See, column 3, lines 24-30. Swank's hash process is very specifically directed to minimizing the amount of data that is sent between a terminal and a host. Specifically, as set forth in column 3, a text file together with its hash is first communicated from a host to a terminal. A non-editable hash file is created at the terminal and contains the host-computed hash value and a terminal-generated hash value for each line of the file. After some editing has occurred to the text file, a hash is re-computed for each line of text and compared line-by-line with the previously-computed has file to ascertain which lines have changed. Once identified, only changed lines are sent to the host. The only thing that Swank and this claim have in common is that the term "hash" is mentioned. Again, this is not the standard that is to be used in making out an obviousness rejection.

Accordingly, for at least this reason, this claim is allowable.

**Claims 48-50** depend from claim 47 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 47, are neither disclosed nor suggested in the references of record, either singly or in combination with one another. In addition, given the allowability of claim 47, the rejection of claim 48 over Iannucci is not seen to add anything of significance.

**Claim 51** recites an updating method for updating software extensions via the Internet comprising:

- receiving, via the Internet, a package manifest containing a list of multiple files that comprise a newer version of a software extension that is to be incorporated into an application program executing on a client that contains an older software extension version, the list containing a hash for one or more of the files comprising the newer version of the software extension;
- comparing one or more hashes that are received with one or more hashes of files from the older version of the software extension;
- for any hashes of corresponding files from the different versions that are different, downloading a new file from a web server; and
- for any hashes of corresponding files from the different versions that are the same, copying a file from an old local directory on the client to a new local directory on the client associated with the newer version of the extension.

In making out the rejection of this claim, the Office argues that Rowley teaches receiving a package manifest and comparing files of an older version with files of a newer version stored in a manifest, and if corresponding files are different, downloading the new files. The Office admits that Rowley does not teach the use of hashes and relies instead on Swank. Relying on Swank, the Office argues that Swank teaches storing a file hash for a file that is to be updated, creating an updated hash of a received file and comparing the hashes so that when the hash of an old file and a newly received file is the same, the old file is updated and replaced by the new file.

The Office further argues that it is possible to use Swank's method to determine which files need to be downloaded to the client. See, e.g. Office Action, page 10, second full paragraph. ***Applicant respectfully submits that the***

***possibility* of using Swank in the manner argued by the Office is not the legally correct obviousness standard.**

Applicant respectfully submits that the Office has misinterpreted this reference. Responsively, the Office argues that it uses Swank simply for the proposition of using hashes to determine changes in files.

According to Swank's system, an original file is maintained at the host and has an associated hash. When changes are made to the file on a remote terminal, those changes are communicated to the host. As specifically taught by Swank, the original file at the host is only updated after a hash value for the original entire file has been recomputed by the host and checked so that it matches the entire file hash total previously stored at the terminal. This ensures that the original file ***at the host*** has not been changed since the terminal hash file was generated. See, column 3, lines 45-50. Thus, Swank teaches only using a hash to determine that an ***old file*** at the host is still, in fact, the ***old file*** from which the terminal's hash was computed. Once confirmed, only those changes transmitted by the terminal are made to the file. The terminal does not transmit the entire changed file—in fact, Swank's subject matter is specifically directed to avoiding any such situation. Thus, at best, Swank simply teaches using a hash to ascertain whether the host's old file has been changed since the terminal file's hash was computed so that the host can incorporate only those changes transmitted to the host by the terminal.

Recharacterized a bit, Swank's approach can be thought of as taking a hash of an old file and if the hash of the old file is the same as the hash of the old file at the terminal, making the changes sent by the terminal. Applicant's claim recites, *inter alia,* "for any hashes of corresponding files from the different versions that

are different, downloading a new file from a web server". Thus, this subject matter is directed to downloading a new file if the hashes are different.

It would appear that Swank is not on point for a couple of different reasons. First, as noted above, Swank is not at all directed to updating software extensions as recited in this claim. Rather, Swank appears to be directed to a non-analogous area of art. Second, Swank's process relied upon by the Office is not directed to comparing hashes for newer versions of anything with older versions of anything for the purpose acquiring a newer version. Rather, Swank's process is directed to comparing such hashes and, if different, possibly acquiring the older version (i.e. the old document) so that the text in the old document can be updated.

The Office's attempted combination is misplaced and, at best, rests on hindsight reconstruction which has been specifically proscribed by the Federal Circuit.

For at least these reasons, the Office has failed to establish a *prima facie* case of obviousness and this claim is allowable.

**Claims 52-54** depend from claim 51 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 51, are neither disclosed nor suggested in the references of record, either singly or in combination with one another. In addition, given the allowability of claim 51, the rejection of claims 52 and 54 over Iannucci is not seen to add anything of significance.

**Claims Rejected Over the Combination that Includes at least Collins and Carpenter**

**Claim 64** recites a queue management method comprising:

- defining a download queue that controls when files are to be downloaded to a client, the files pertaining to a software extension that is to be incorporated into an application program executing on the client;
- ascertaining whether a user action at the client requires one or more files that are not currently being downloaded; and
- manipulating the download queue responsive to a user action that requires one or more files that are not currently being downloaded so that the one or more required files are downloaded sooner than they would otherwise be.

In making out the rejection of this claim, the Office argues that Collins teaches a download queue that controls when files are to be downloaded to a client. The Office admits that Collins does not teach ascertaining whether a user action at the client requires one or more files that are not currently being downloaded, and manipulating the download queue responsive to a user action that requires one or more files that are not currently being downloaded so that the one or more required files are downloaded sooner than they would otherwise be.

The Office then relies on Carpenter and argues that Carpenter teaches queue manipulation based on user input, citing to column 7, lines 21-25. Based on this, the Office argues that the subject matter of this claim would be obvious in view of Collins and Carpenter reasoning that such would allow a user to prioritize files that need to be downloaded while the queue is in progress. Further, the Office addresses Applicant's previous arguments with respect to Carpenter and notes that Carpenter was "introduced to show that a feature of a queue could be user manipulation of the queue". See, e.g. Office Action page 19 second full paragraph.

Applicant respectfully disagrees and submits that the Office has taken Carpenter out of context. Specifically, Carpenter's system and method are

designed to operate for transmitting messages from a memory constrained data processor to a host computer. See, e.g. column 4, lines 47-49. That is, in Carpenter, it is the client device that is memory constrained and transmission occurs *from* the client *to* the host. Thus, in Carpenter the queue management referred to by the Office (column 7, lines 21-25) takes place with respect to transmissions made from the client. Put another way, the queue management does not take place with respect to the client downloading anything. This is particularly germane when the claim language of this claim is examined.

Specifically, the claim recites that the download queue is defined and controls when files are to be *downloaded* to a client. The remainder of the claim (the recited acts of ascertaining and manipulating) are directed to ascertaining whether a user action requires one or more files that are not currently being downloaded and if so, manipulating the download queue so that one or more required files are downloaded sooner than they would otherwise be. Hence, Carpenter fails to teach queue management as recited in this claim. Specifically, Carpenter fails to teach the subject matter missing from Collins III—i.e. "ascertaining whether a user action at the client requires one or more files that are not currently being downloaded, and manipulating the download queue responsive to a user action that requires one or more files that are not currently being downloaded so that the one or more required files are downloaded sooner than they would otherwise be". Carpenter's scant teaching of user manipulation of a queue falls far short of supplying this missing subject matter and a motivation to use it to modify Collins III.

Accordingly, for at least this reason, the Office has failed to establish a *prima facie* case of obviousness and this claim is allowable.

**Claims 65-68** depend from claim 64 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 64, are neither disclosed nor suggested in the references of record, either singly or in combination with one another. In addition, given the allowability of claim 64, the rejection of claim 66 over the combination with Van Huber is not seen to add anything of significance.

**Claims Rejected Over the Combination of at least Halpern and Taylor**

**Claim 69** recites a method of creating software packages for delivery via the Internet comprising:

- identifying end user features;
- identifying shared dependencies between the end user features;
- creating individual software packages for the end user features;
- creating individual software packages for the shared dependencies; and
- hosting the software packages on a web server;
- wherein both of said acts of identifying and both of said acts of creating provide software extensions that are created in a uniform manner independent of end user input.

In making out the rejection of this claim, the Office argues that Halpern teaches identifying end user features, creating individual software packages and hosting the software packages on a web server. The Office then relies on Taylor and argues that it teaches identifying shared dependencies between end user features and creating individual software packages for the shared dependencies. Based on the teachings of these two references, the Office argues that the subject

matter of this claim is obvious. Applicant previously disagreed with the Office's position and continues to disagree.

Nonetheless, Applicant previously made a clarifying amendment to this claim that both of the acts of identifying and both of the acts of creating provide software extensions that are created in a *uniform manner independent of end user input*. Specifically, the subject matter of this claim is directed to embodiments that enable packages to be created in a uniform manner in order to provide an organized delivery process. As noted in the Specification, one of the features of the described embodiment pertains to its extensibility. More specifically, software packages can be created by third party developers for extending so-called software platforms. See, e.g. Specification, page 49, lines 10-14 ("The embodiments described above provide a platform solution that provides for customization and extensibility through a consistent and logical extensibility mechanism and object model that can be easily understood by third party developers. Internet-based downloads can be accomplished without a great deal of user intervention and without manipulating any user persisted settings.").

Halpern, as Applicant previously noted, is very specifically directed to building installation packages that necessarily *require* user input. See, e.g. Abstract ("An installation package delivered to a requesting end user is custom configured at a remote server...in response to the user's inputs); column 5, lines 49-51 ("In response to the user's selections, the options manager 104 delivers an installation and/or options specification to an installer set generator 109."); and column 7, lines 22-23 ("Step 3: The user selects desired software components and options from a list of components and options.").

In responding to Applicant's previous amendment and argument, the Office states, citing for support to *In re Venner,* 262 F.2d 91, 95, 120 USPQ 193, 194 (CCPA 1958), that "[a]lthough neither Halpern nor Taylor teach that the acts of identifying and creating are performed without user input, the act of automating an otherwise manual process is obvious and does not distinguish over the prior art."

The logic of this argument is flawed for at least this following reason. This argument completely disregards the context of both Applicant's claimed subject matter and the cited reference, and establishes what amounts to a *per se* rule that any automation is not patentable. Applicant is unaware of any sections of the Patent Statute that establish a rule that "automation", as referred to by the Office, constitutes unpatentable subject matter. It is interesting to note that the Board of Patent Appeals and Interferences has negatively treated *In re Venner*. In addition, the Federal Circuit has expressed *per se* rules of obviousness as legally incorrect.

Specifically, in *Ex Parte Richard Brouillet, Jr.*, 2001 WL 1339914 (2001), the Board distinguished *In re Venner* because in *In re Venner*, all of the limitations in the claims, including the automatic means, were disclosed in the applied references. The prior art in *Ex Parte Richard Brouillet, Jr.* was missing a claim element. The Board then quotes the Federal Circuit case of *In re Ochiai*, 71 F. 3d 1565, 1572, 37 USPQ2d 1127, 1133 (1995) in stating that "reliance on *per se* rules of obviousness is legally incorrect and must cease."

The Office admits that neither Halpern nor Taylor teach that the acts of identifying and creating are performed without user input. It is inappropriate for the Office to reject as obvious the subject matter of this claim using nothing more than a *per se* rule of obviousness, when such *per se* rules have been indicated by the Federal Circuit to be legally incorrect.

Accordingly, for at least this reason, the Office has failed to establish a *prima facie* case of obviousness and this claim is allowable.

**Claim 70** depends from claim 69 and is allowable as depending from an allowable base claim. This claim is also allowable for its own recited features which, in combination with those recited in claim 69, are neither disclosed nor suggested in the references of record, either singly or in combination with one another. In addition, given the allowability of claim 69, the rejection of this claim over the combination with Rowley is not seen to add anything of significance.

**Claims Rejected Over the Combination of at least Bailey and Kolawa**

**Claim 86** recites a method of providing software extensions via the Internet comprising:

- assigning one or more files to one or more scenarios to provide multiple different scenarios that describe ways that a user interacts with a software application program;
- assigning a priority to each of the scenarios;
- sorting multiple files in accordance with their scenario priority or priorities; and
- downloading sorted files in an order defined by said sorting.

In making out the rejection of this claim, the Office argues that Bailey teaches running a number of test scenarios on a file which represents a program (citing to column 1, lines 22-28 and lines 46-50), and assigning a priority to tests by ordering them based on coverage. The Office then goes on to argue that Bailey does not explicitly teach that the scenarios describe ways in which a user interacts with an application. The Office then relies on Kolawa and argues that it teaches a test suite which is a collection of tests that test the complete functionality of a

software program. Based on this, the Office surmises that Kolawa *must* describe ways that a user can use an application.

Continuing, the Office admits that neither Bailey nor Kolawa teach sorting multiple files in accordance with their scenario priority or downloading sorted files in an order defined by the sorting. The Office then relies on Collins and argues that it teaches such subject matter in column 5, lines 35-37. Based on the teachings of these three references, the Office argues that the subject matter of this claim is obvious. Applicant respectfully disagrees and submits that the Office has not established a *prima facie* case of obviousness for a number of different reasons.

First, Bailey discloses a *software testing system* that measures execution of machine code instructions in an executing program. The first two sections of Bailey cited by the Office simply describe the importance of comprehensively testing software and a particular type of tool that has been used to test software respectively. The third section of Bailey that is cited by the Office describes part of its technique for measuring the execution of machine code instructions in a computer program. It appears, from even a cursory reading of Bailey, that Bailey is not remotely associated with or concerned with methods of providing software extensions via the Internet. Thus, to this extent, Applicant continues to maintain that Bailey is not even germane to the subject matter of this claim.

The reference to Kolawa is no better. Kolawa is also directed to software testing and discloses a system and method that generates a test suite for a computer program that comprises program statements and variables including at least one input statement having one or more input variables that are grouped into code blocks and stored in a program database. The program statements

corresponding to a candidate code block are read from the program database and each input variable is represented in symbolic form as a symbolic memory value. The processing that Kolawa further describes makes it abundantly clear that Kolawa is neither directed to nor in any way concerned with methods of providing software extensions via the Internet.

In responding to Applicant's previous arguments, the Office states that Bailey and Kolawa are introduced to teach assigning files to scenarios that describe ways a user interacts with a software application program and assigning priorities to the scenarios. The Office then states that "Collins teaches placing software packages in a queue for downloading, thus sorting the packages in an order for downloading, where this order is taught in Bailey as a certain scenario priority order."

Applicant respectfully submits that the Office has engaged in impermissible hindsight reconstruction which has been specifically proscribed by the Federal Circuit. Based on the substantial differences between the subject matter of the present claim and the references to Bailey and Kolawa, the Office has failed to establish a *prima facie* case of obviousness. Accordingly, for at least this reason, this claim is allowable. In addition, based on the failure of the Office to establish a *prima facie* case of obviousness, the Office's reliance on Collins is not seen to add anything of significance. Accordingly, this claim is allowable.

**Claims 87-90** depend from claim 86 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 86, are neither disclosed nor suggested in the references of record, either singly or in combination with one another.

**Claims Rejected Over the Combination of at least Rowley and Collins**

**Claim 91** recites a method of ordering files for download to a client comprising:

- sorting multiple files by one or more file groups;
- sorting the multiple files based on scenario priority of one or more scenarios into which each file can be placed;
- sorting the multiple files by file usage order within one or more scenarios.

In making out the rejection of this claim, the Office argues that Rowley teaches sorting multiple files into multiple directories, but not sorting multiple files based on scenario priority. The Office then relies on Collins and argues that it teaches sorting data packages on a queue for downloading and hence prioritizing certain packages. The Office then admits that neither Rowley nor Collins teach sorting multiple files based on file usage order. The Office then relies on Bailey and argues that it teaches ordering scenarios by priority, citing to column 8, lines 21-27. Based on the teachings of these three references, the Office argues that the subject matter of this claim is obvious. Applicant respectfully disagrees and submits that the Office has not established a *prima facie* case of obviousness.

Specifically, Collins simply discloses that once a software package is scheduled for transmission via the internetwork to a target computer, group or Profile, an indication is stored in the Outbound Package Queue (13). See, column 5, lines 34-37. Applicant respectfully submits that this in no way describes or suggests "sorting the multiple files based on ***scenario priority*** of one or more ***scenarios into which each file can be placed***.

As but one example of subject matter that is covered by this claim, the Office is respectfully referred to the Specification, page 31, line 1 through page 33, line 13. As should be readily apparent after this example is considered, Collins in no way discloses or suggests "sorting the multiple files based on ***scenario priority*** of one or more ***scenarios into which each file can be placed***."

The Office seems to agree with Applicant on this point and argues that Bailey teaches sorting scenarios by priority. See, e.g. Office Action page 20, last paragraph. Applicant respectfully submits that this claim recites no such subject matter. That is, this claim recites:

- ***sorting multiple files*** by one or more file groups;
- ***sorting the multiple files*** based on scenario priority of one or more scenarios into which each file can be placed;
- ***sorting the multiple files*** by file usage order within one or more scenarios.

Accordingly, for at least this reason, the Office has failed to establish a *prima facie* case of obviousness and this claim is allowable.

**Claims 92-96** depend from claim 91 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 91, are neither disclosed nor suggested in the references of record, either singly or in combination with one another.

**Conclusion**

Applicant has made a sincere attempt to place this application in condition for allowance and respectfully requests a Notice of Allowability be issued

forthwith. If the next anticipated action is to be anything other than issuance of a Notice of Allowability, Applicant respectfully requests a telephone call for the purpose of discussing an Appeal.

Respectfully Submitted,

Dated: 6/28/04

By: _____

Lance R. Sadler
Reg. No. 38,605
(509) 324-9256